Programming Fundamentals

Self-Assessment Exercises

Please Read Following very carefully

1. **FIRST MOST IMPORTANT: DO NOT DISCUSS** or **COPY** any code or logic from your pairs. This is **UN GRADED SELF ASSESSMET** that shall help both the students and instructors to work **TOGETHER** to bridge the **GAP** if there is ANY.
2. **SECOND MOST IMPORTANT: FIRST** write an **ALGOITHM** and draw **FLOWCHART** in your Register and then write and compile **CODE**.
3. **THIRD MOST IMPORTANT THING:** Write all the code in the form of functions.
4. This level covers the following Course Learning Outcome of the Course
    a. **CLO 1:** Student should be able to design programming logic and write code for complex problems using Computational Steps.

5. These exercise shall help you to quantify your level of programming for 6 **LOs** and give you directions to work on the weak areas. These exercises cover all the concepts that includes the fundamental programming concepts: Input, Output, Variables, Data types, IF-Else block, Conditional Operators, Logical Operators, For Loop and While Loop, arrays, function and file handling.

6. To achieve CLO 1, we need to pass through different Learning Objectives. These learning objectives include following:

LO 1	Student should be able to take input from console, store the input in variables, apply arithmetic operators on the variables and give output at the console.
LO 2	Students should be able to write complex conditional statements using multiple logical operators.
LO 3	Student should be able to solve the complex problems using loops and conditional statements.
LO 4	Student should be able to solve complex problems by dividing into smaller reusable blocks of code
LO 5	Student should be able to solve complex problems by taking large amount of data into arrays
LO 6	Student should be able to store and manipulate data that is stored permanently.

Name _____ **Roll Number** _____

Variables and conditional Statements	Task	How I AM Doing the Programming				How I AM Feeling			
		I did it without any help	I did it with a little help.	I made the logic but my code is not working	I could not make the logic. I need more help to understand it	**Confident!** It was Piece of Cake.	**Thrilled!** It was challenging but I am feeling Thrilled Now.	**Confused!** I get confused after this exercise	**Depress!** I am Depress because I cannot do it.
	L 1.1								
	L 1.2								
	L 1.3								
	L 1.4								
	L 1.5								
Loops and Arrays	**L 2.1**								
	L 2.2								
	L 2.3								
	L 2.4								
File Handling	**L 3.1**								
	L 3.2								
	L 3.3								
	L 3.4								
	L 3.5								

Self-Assessment Tasks

L 1.1:

A tetrahedron is a pyramid with a triangular base and three sides. A tetrahedral number is a number of items within a tetrahedron.
Create a function that takes an integer n and returns the nth tetrahedral number.

There is a formula for the nth tetrahedral number.

$T_n = (n * (n + 1) * (n + 2)) / 6$

Test Cases:

```
tetra(2) → 4
tetra(5) → 35
tetra(6) → 56
```

L 1.2:

In a board game, a piece may advance 1-6 tiles forward depending on the number rolled on a six-sided dice. If you advance your piece onto the same tile as another player's piece, both of you earn a bonus.
Can you reach your friend's tile number in the next roll? Create a function that takes your position a and your friend's position b and returns a boolean representation of whether it's possible to earn a bonus on any dice roll.

Notes

- You cannot move backward (which is why example #3 doesn't work).
- If you are already on the same tile, return `false`, as you would be advancing away.
- Expect only positive integer inputs.

Test Cases:

```
possibleBonus(3, 7) → true
possibleBonus(1, 9) → false
possibleBonus(5, 3) → false
```

L 1.3:

It's a Pokemon battle! Your task is to calculate the damage that a particular move would do using the following formula (not the actual one from the game):

damage = 50 * (attack / defense) * effectiveness

- attack = your attack power
- defense = the opponent's defense
- effectiveness = the effectiveness of the attack based on the matchup (see explanation below)

Effectiveness:

Attacks can be super effective, neutral, or not very effective depending on the matchup. For example, water would be super effective against fire, but not very effective against grass.

Super effective: 2x damage
Neutral: 1x damage
Not very effective: 0.5x damage

To prevent this challenge from being tedious, you'll only be dealing with four types: fire, water, grass, and electric. Here is the effectiveness of each matchup:

- fire > grass
- fire < water
- fire = electric
- water < grass
- water < electric
- grass = electric

The function you must implement takes in:

- your type
- the opponent's type
- your attack power
- the opponent's defense

Note:

Any type against itself is not very effective. Also, assume that the relationships between different types are symmetric (if A is super effective against B, then B is not very effective against A).

Test Cases:

```
calculateDamage("fire", "water", 100, 100) → 25
calculateDamage("grass", "fire", 35, 5) → 175
calculateDamage("electric", "fire", 100, 100) → 50
```

L1.4:

A Fruit shop sell fruits on at different prices on WeekDays and on at Week Ends according to following price list

Fruits	**Week Days**	**Week Ends**
Banana	**2.50**	**2.70**

L 1.5:

Write a program that checks whether a point {x, y} is placed onto any of the sides of a rectangle {x1, y1} – {x2, y2}.

Apple	**1.20**	**1.25**
Orange	**0.85**	**0.90**
Grapefruit	**1.45**	**1.60**
Kiwi	**2.70**	**3.00**
Pineapple	**5.50**	**5.60**
Grapes	**3.85**	**4.20**

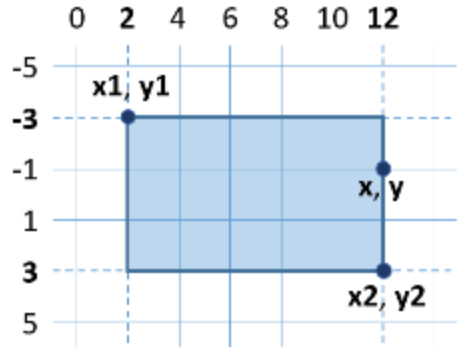
Input:
a day of the week (Monday / Tuesday / …)
quantity: (a decimal number)

Output:

Calculates the price according to the prices from the tables above.

Print “error” if it is an invalid day of the week or an invalid name of a fruit.

Test Case 01 **Input:** **Fruit: Orange** **Day : Sunday** **Qty : 3** **Output: 2.70**	**Test Case 02** **Input:** **kiwi** **Monday** **2.5** **Output: 6.75**
Test Case 03	**Test Case 04**



Input

The input data is read from the console and consists of 6 lines: the decimal numbers x1, y1, x2, y2, x and y (as it is guaranteed that x1 < x2 and y1 < y2)

Output

Print "Border" (if the point lies on any of the sides) or "Inside / Outside" (in the opposite case)

Input	**Output**
2 **-3** **12** **3** **12** **-1**	**Border**

Input	**Output**

Input: **Grapes** **Saturday** **0.5** **Output: 2.10**	**Input:** **tomato** **Monday** **0.5** **Output: Error**

2 **-3** **12** **3** **12** **-1**	**Inside/ OutSide**

L 2.1

To train for an upcoming marathon, Johnny goes on one long-distance run each Saturday. He wants to track how often the number of miles he runs this Saturday exceeds the number of miles run the previous Saturday. This is called a progress day.

Create a function that takes in an array of miles run every Saturday and returns Johnny's total number of progress days.

Note:

Running the same number of miles as last week does not count as a progress day.

Test Cases:

- `[3, 4, 1, 2]`
  `progressDays()` → `2`
  `// There are two progress days, (3->4) and (1->2)`
- `[10, 11, 12, 9, 10]`
  `progressDays()` → `3`
- `[6, 5, 4, 3, 2, 9]`
  `progressDays()` → `1`
- `[9, 9]`
  `progressDays()` → `0`

L 2.2

Mubashir needs your help to count a specific digit in an array.

You have to create a function that takes two integers n and d and makes an array of squares of all numbers from 0...<= n and returns the count of the digits d in the array.

Note:

d will always be 0<=d<10.

Test Cases:

`countDigits(10, 1)` → `4`

L 2.3

A long stretch of beach is represented by a string of two characters 0 - free, 1 - occupied. Due to recent restrictions, a new person cannot take place next to another. There has to be one free place between two people lounging on the beach. Create a function to compute how many new people at most can settle in on the given beach.

Test Cases:

`sunLoungers("10001")` → `1`

```
// Squared array from 0 to 10 =
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
// Digit 1 appeared 4 times in the array

countDigits(25, 2) → 9

countDigits(10, 1) → 4
```

```
// Can take place in the middle.
sunLoungers("00101") → 1
// Can take place on the left.
sunLoungers("0") → 1
// Can take one place.
sunLoungers("000") → 2
// Can take places on the left and on the right.
```

L 2.4

At a lemonade stand, each lemonade costs $5. Customers are standing in a queue to buy from you, and order one at a time (in the order specified by bills). Each customer will only buy one lemonade and pay with either a $5, $10, or $20 bill. You must provide the correct change to each customer so that the net transaction is that the customer pays $5.

Return true if and only if you can provide every customer with correct change.

Note:

You don't have any change in hand at first.

bills is an integer array.

bills[i] will be either 5, 10, or 20.

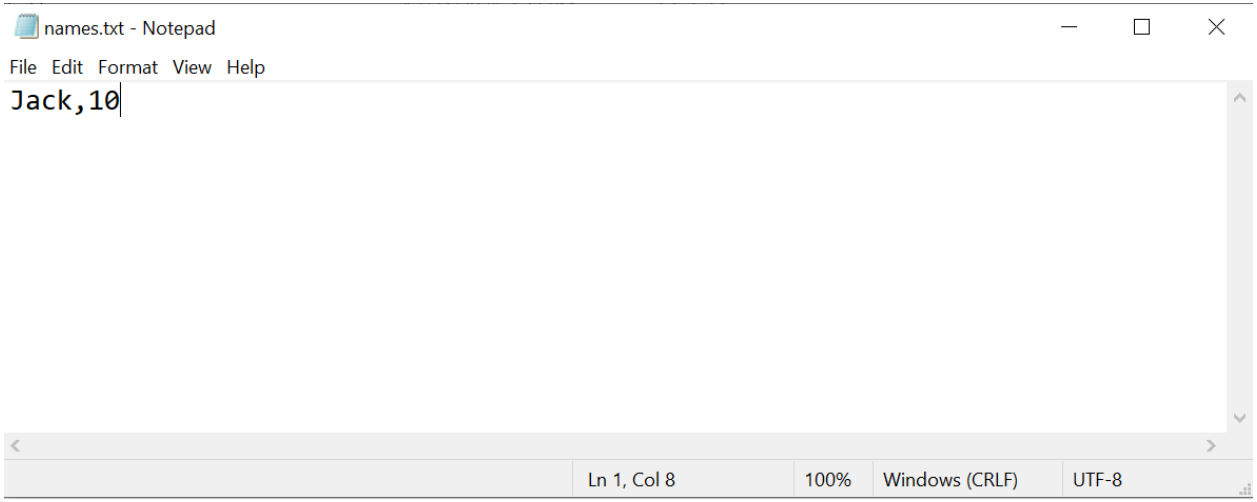
Test Cases:

- `[5, 5, 5, 10, 20]`
  `lemonade() → true`
- `[5, 5, 10, 10, 20]`
  `lemonade() → false`
- `[10, 10]`
  `lemonade() → false`
- `[5, 5, 10]`
  `lemonade() → true`

L 3.1

Create a function which constructs a rectangular birthday cake, based on someone's name and age!

Name and age is given in a text file named names.txt in the following format.



Read the file to get the name and age of the person and make sure to surround the birthday message with the character that fits the rule:

1. If the age is an even number, surround the message with "#".
2. If the age is an odd number, surround the message with "*".

Store the output in another file.

Other important rules:

1. The output message should be in the format:

{age} HB {name}! {age}

1. Leave a space between the edge of the cake and the age numbers.
    1. The number of characters (# or *) in the banner should be 15

Test Cases

Input	Output
getBirthdayCake("Jack", 10)	"################" "# 10 HB Jack! 10 #" "################"
getBirthdayCake("John", 19)	"***************" "* 19 HB John! 19 *" "***************"
getBirthdayCake("Mary", 20)	"################"

		"# 20 HB Mary! 20 #" "################"

L 3.2

Create a function that takes a string str containing only letters from a to z from a file named **alphabets.txt** in lowercase and appends the missing letter(s) in alphabetical order a-z in the next line of the file.

A set of letters is given by

abcdefghijklmnopqrstuvwxyz

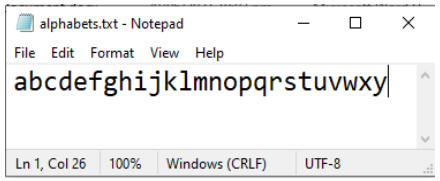
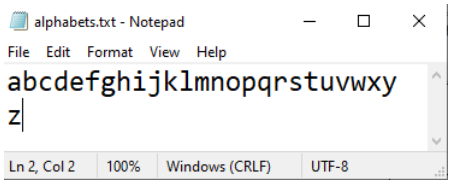
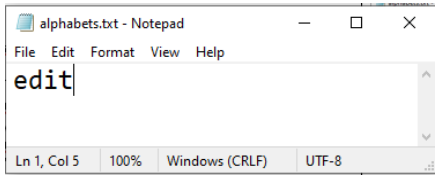
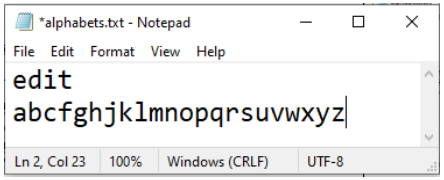
Notes

If the string contains all letters from a-z, then an empty string will be appended.

Hint: You can convert int to char by writing

char(97) → a

Test Cases

Input	Output
	
	

L 3.3

Suppose you are working for Secret Services. You have to write a function that takes a string and convert that into morseCode. The function outputs an encrypted letter string with Morse code.

Note:

- There will be a space in the output string after each letter's morse code.

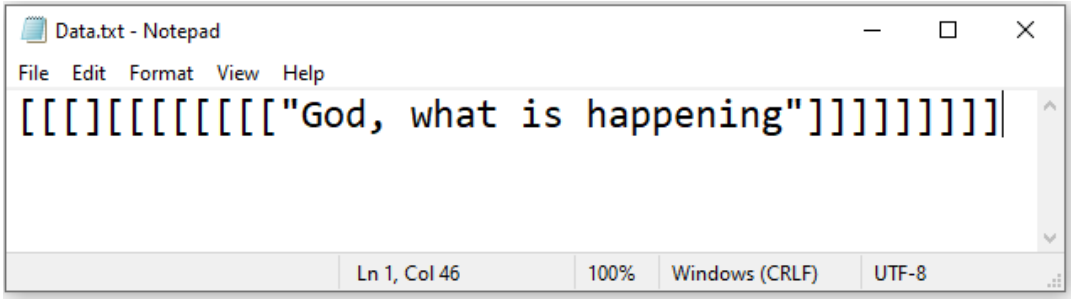
Alphabets	MorseCode
A	.-
B	-...
C	-.-.
D	-..
E	.
F	..-.
G	--.
H	....
I	..
J	.---
K	-.-
L	.-..
M	--
N	-.
O	---
P	.--.
Q	--.-
R	.-.
S	...
T	-

U	..-
V	...-
W	.--
X	-..-
Y	-.--
Z	--..
(Space)	-.-.-.-

Test Cases

Input	Output
morse("F Mueller")	"..-. ..... -- ..- . .-.. .-.. . .-."
morse("Barack Obama")	"-... .- .-. .- -.-. -.- ..... --- -... .- -- .-"
morse("JOHN F KENNEDY")	".--- --- .... -. ....-. ..... -.- . -. -. . -.. -.--"

L 3.4
John has made a mistake and overwrote a text file with some weird data, thankfully he had no back-up and he's too lazy to fix it so now you guys can solve it for him!
Get the text strings out of those files.

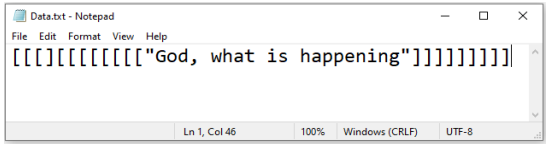
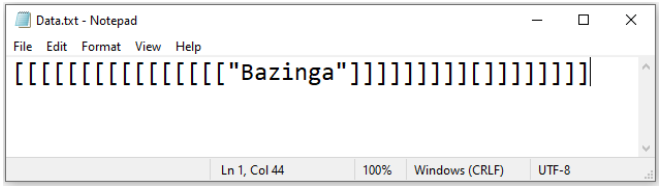
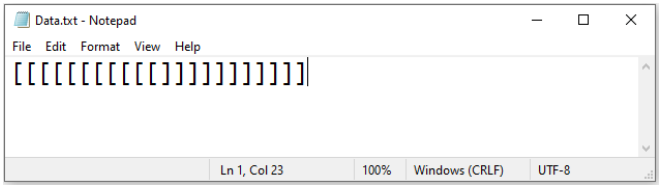


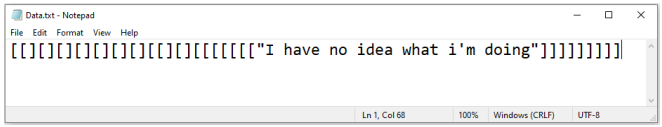
Note:

- if there *is no string*, you should return a pre-defined string (see Test Case 3).

Your task is to make a function that reads the **data.txt** file and then returns only the text string from the data.

Test Cases

Input	Output
	"God, what is happening"
	"Bazinga"
	"What... why did you make this?"

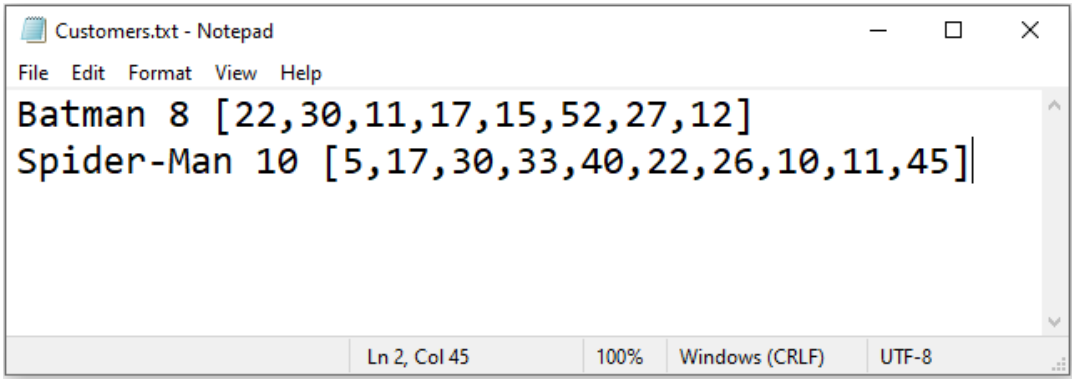
		"I have no idea what i'm doing"

L 3.5

KamyabLife is launching a network of autonomous pizza delivery drones and wants you to create a flexible rewards system (Pizza Points) that can be tweaked in the future. The rules are simple:

if a customer has made at least N orders of at least Y price, they get a FREE pizza!

The information of the customers is stored in a file **Customers.txt** in the following format. First the name of the customer is given then after the space the number of orders are given then after the space within the brackets all the orders prices are given.



Your task is to create a function that takes a minimum number of orders and a minimum order price then cout the names of the customers that are eligible for a free pizza.

Test Cases

Input	Output
pizza_points(5, 20)	"Spider-Man"
pizza_points(3, 10)	"Batman" "Spider-Man"

	pizza_points(5, 100)	“”	